

A Categorical Model for the Lambda Calculus with Constructors

Barbara Petit

Focus - INRIA - Università di Bologna
barbara.petit@ens-lyon.org

Abstract:

The lambda calculus with constructors is an extension of the lambda calculus with variadic constructors. It decomposes the pattern-matching à la ML into a case analysis on constants and a commutation rule between case and application constructs. Although this commutation rule does not match with the usual computing intuitions, it makes the calculus expressive and confluent, with a rather simple syntax. In this paper we define a sound notion of categorical model for the lambda calculus with constructors. We then prove that this definition is complete for the fragment of the calculus with no match-failure, using the model of partial equivalence relations.

KEYWORDS: Lambda calculus, Pattern matching, Semantics, Categorical model, PER model.

Introduction

Pattern matching is now a key feature in most functional programming languages. Inherited from the simple constants recognition mechanism that appeared in the late 60's (in *Snobol* or in *Pascal* for instance), it is now a elaborated feature in main programming languages (*ML*, *Haskell* etc.) and some proof assistants (such as *Coq* or *Agda*), able to decompose complex data-structures.

Its theoretical aspects are being intensively studied since the 90's [5, 11]. In particular, several lambda calculi with pattern matching have been proposed [19, 4, 8]. Among them, the lambda calculus with constructors [1] (or $\lambda_{\mathcal{C}}$ -calculus) offers the advantage of having simple computation rules. Indeed, the pattern matching à la ML is there decomposed into two atomic rules (a constants analysis rule, and a commutation rule). The rather simple syntax of this calculus together with the decomposition of its powerful computational behaviour into elementary steps stimulate a semantic study of the $\lambda_{\mathcal{C}}$ -calculus from a categorical point of view.

As far as we know, no categorical model had been proposed so far for a calculus with pattern matching. Yet category theory allows to express some generic semantic properties on a calculus, and to factorise many of its different concrete models. Furthermore, when the categorical model is *complete*, it synthesises exactly the extensional properties of the calculus. Since the description of the models for the pure lambda calculus as Cartesian closed categories with a reflexive object [16], some complete categorical models have been defined for variants of the lambda calculus [7, 17, 6].

In this paper, after a brief presentation of the $\lambda_{\mathcal{C}}$ -calculus (Sec.), we establish a categorical definition of models for it (Sec. 2). We then prove that it is to some extent complete for the

$\lambda_{\mathcal{C}}$ -calculus, using the standard PER model and some rewriting techniques (Sec. 3). Notice that we only use very basic notions of category theory (knowledge of the first two chapters of [3] is sufficient).

1 The lambda calculus with constructors

The lambda calculus with constructors extends the pure lambda calculus with pattern matching features: a set of constants (that we consider here to be finite of cardinal n) called *constructors* and denoted by \mathbf{c}, \mathbf{d} etc. is added, with a simple mechanism of case analysis on these constants (similar to the **case** instruction of Pascal):

$$\{\mathbf{c}_1 \mapsto t_1; \dots; \mathbf{c}_k \mapsto t_k\} \cdot \mathbf{c}_i \rightarrow t_i \quad (\text{CASECONS})$$

Although only constant constructors can be analysed, a matching on variant constructors can be performed *via* a commutation rule between case construction and application:

$$\{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t) u \quad (\text{CASEAPP})$$

This commutation rule enables simulating any pattern matching *à la* ML, by generalising the following example: in the $\lambda_{\mathcal{C}}$ -calculus, the predecessor function on unary integers (represented with the constructors 0 and S) is implemented as $\text{pred} = \lambda x. \{\mathbf{0} \mapsto 0; \mathbf{S} \mapsto \lambda y. y\} \cdot x$. Applying this function to a non zero integer $\mathbf{S} n$ actually produces the expected result:

$$\begin{aligned} \text{pred} (\mathbf{S} m) &\rightarrow \{\mathbf{0} \mapsto 0; \mathbf{S} \mapsto \lambda y. y\} \cdot (\mathbf{S} m) \\ &\rightarrow (\{\mathbf{0} \mapsto 0; \mathbf{S} \mapsto \lambda y. y\} \cdot \mathbf{S}) m \rightarrow (\lambda y. y) m \rightarrow m \end{aligned}$$

Formally, the syntax of the $\lambda_{\mathcal{C}}$ -calculus is defined by the following grammar:

$$\begin{aligned} t, u, v &:= x \mid tu \mid \lambda x. t \mid \mathbf{c} \mid \{\theta\} \cdot t \\ \theta, \phi &:= \{\mathbf{c}_1 \mapsto u_1; \dots; \mathbf{c}_k \mapsto u_k\} \quad (\text{with } k \geq 0 \text{ and } \mathbf{c}_i \neq \mathbf{c}_j \text{ for } i \neq j) \end{aligned}$$

In the terms (denoted by t, u etc.) the application takes precedence over lambda abstraction and case construct. Notice that constructors, like any terms, can be applied to any number of arguments and thereby are *variadic* (they have no fix arity). We call *data-structure* a term on the form $\mathbf{c} t_1 \dots t_k$.

A *case-binding* θ is just a (partial) function from constructors to terms, whose domain is written $\text{dom}(\theta)$. By analogy with sequential notation, we may write $\theta_{\mathbf{c}}$ for u when $\mathbf{c} \mapsto u \in \theta$. In order to ease the reading, we may write $\{\mathbf{c}_1 \mapsto u_1; \dots; \mathbf{c}_n \mapsto u_n\} \cdot t$ instead of $\{\{\mathbf{c}_1 \mapsto u_1; \dots; \mathbf{c}_n \mapsto u_n\}\} \cdot t$. The usual definition of the free variables of a term is naturally extended to the new constructions of the calculus, taking care that constructors are not variables (and therefore not subject to substitution nor α -conversion).

In this calculus, a *match failure* is a term $\{\theta\} \cdot \mathbf{c}$ where $\mathbf{c} \notin \text{dom}(\theta)$. We say that a term is *defined* when none of its subterm is a match failure, and that it is *hereditarily defined* when all this reduces (in any number of steps, including zero) are defined.

Reduction rules are given in Fig. 1. In addition to the usual β -reduction (called APPLAM) and to the two rules presented earlier, there is a rule of commutation between case construct and lambda abstraction (CASELAM) to ensure confluence [1, Cor. 1], and the usual η -reduction (called LAMAPP) as well as a rule of composition of case-bindings (CASECASE) so that the calculus enjoys the *separation property* [1, Theo. 2]. More explanations and examples about this calculus can be found in [2, 12].

APPLAM	(AL)	$(\lambda x.t)u \rightarrow t[x := x]u$	
LAMAPP	(LA)	$\lambda x.tx \rightarrow t$	$(x \notin \text{fv}(t))$
CASECONS	(CO)	$\{\theta\} \cdot c \rightarrow t$	$((c \mapsto t) \in \theta)$
CASEAPP	(CA)	$\{\theta\} \cdot (tu) \rightarrow (\{\theta\} \cdot t)u$	
CASELAM	(CL)	$\{\theta\} \cdot \lambda x.t \rightarrow \lambda x.\{\theta\} \cdot t$	$(x \notin \text{fv}(\theta))$
CASECASE	(CC)	$\{\theta\} \cdot \{\phi\} \cdot t \rightarrow \{\theta \circ \phi\} \cdot t$	
with $\theta \circ \{\mathbf{c}_1 \mapsto t_1; \dots; \mathbf{c}_n \mapsto t_n\} = \{\mathbf{c}_1 \mapsto \{\theta\} \cdot t_1; \dots; \mathbf{c}_n \mapsto \{\theta\} \cdot t_n\}$			

Figure 1: Reduction rules for $\lambda_{\mathcal{E}}$.

2 The categorical model

In this section we may define a notion of a categorical model for the $\lambda_{\mathcal{E}}$ -calculus, that we prove to be sound. No deep knowledge in category theory is assumed from the reader, he might just know the definition of a Cartesian closed category (also said a CCC).

The notations we use are quite standard: in a CCC, the product of two objects A and B is written $A \times B$ and their exponential B^A . The k -ary product of A is denoted by A^k , and the identity morphism on A by Id_A (or simply Id if it raises no ambiguity). The i^{th} projection morphism of a k -ary product is written π_i^k , or π_i if $k = 2$. Given some morphisms $f : A \rightarrow B$, $g : A \rightarrow C$ and $h : A \rightarrow C$, $\langle f; g \rangle$ denotes the pairing of f and g , and $f; h$ the composition of f and h . The evaluation map of A and B is $\text{ev} : B^A \times A \rightarrow B$ and the curried form of a morphism f is written $\Lambda(f)$.

2.1 $\lambda_{\mathcal{E}}$ -models

It is well known [10] that Cartesian closed categories have exactly the good structure to interpret the typed lambda calculus. To cope with the problem of self application of terms, such a category must be provided with a reflexive object D in order to interpret the *untyped* lambda calculus [16]. Terms are then interpreted by points of D . The denotation of applications is constructed with a morphism $\text{app} : D \rightarrow D^D$, and the one of lambda abstractions with a morphism $\text{lam} : D^D \rightarrow D$. Also the correction of the β -reduction is ensured by the equality $\text{lam}; \text{app} = Id_{D^D}$ (if moreover $\text{app}; \text{lam} = Id_D$, then the model satisfies the η -equivalence).

Building a model for the $\lambda_{\mathcal{E}}$ -calculus requires some extra morphisms and equalities for the new constructions and the new rules of the calculus. In particular, writing $\{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ the set of constructors, a special point c_i^* of D is needed for each $i \leq n$ to interpret them. The denotations of case-bindings are then points of D^n . A case binding θ is interpreted by the n -tuple $\langle d_1; \dots; d_n \rangle$ where d_i is the denotation of θ_{c_i} if $c_i \in \text{dom}(\theta)$, and is a special point ζ representing match failure otherwise. In order to interpret case constructs, we need a morphism $\text{case} : D^n \times D \rightarrow D$, that transforms the denotation of θ and t into the one of $\{\theta\} \cdot t$.

Let us informally confuse terms and their denotations, and write a case-binding $\{c_i \mapsto u_i / 1 \leq i \leq n\}$ as $\{\vec{c} \mapsto \vec{u}\}$ and its denotation as \vec{u} . Then the rule CASECONS is valid if $\{\vec{c} \mapsto \vec{u}\} \cdot c_i$ and u_i have the same denotation, *i.e.* intuitively if $\text{case}(\vec{u}, c_i) = \pi_i^n(\vec{u})$. This is formally expressed by the commutation of the diagram (D2) in Fig. 2.

In the same way, the rule CASEAPP is valid if the diagram (D3) commutes, *i.e.* if

$$\begin{array}{ccccccc}
D^n \times D \times D & \xrightarrow{-\times \mathbf{app} \times -} & D^n \times D^D \times D & \xrightarrow{-\times \mathbf{ev}} & D^n \times D & \xrightarrow{\mathbf{case}} & D \\
(\vec{u}, t, t') & & (\vec{u}, \hat{x}.tx, t') & & (\vec{u}, tt') & & \{\vec{c} \mapsto \vec{u}\} \cdot (tt')
\end{array}$$

(where $\hat{x}.v$ represents the function mapping v_0 to $v[x := v_0]$) is equal to

$$\begin{array}{ccccccc}
D^n \times D \times D & \xrightarrow{\mathbf{case} \times -} & D \times D & \xrightarrow{\mathbf{app} \times -} & D^D \times D & \xrightarrow{\mathbf{ev}} & D \\
(\vec{u}, t, t') & & (\{\vec{c} \mapsto \vec{u}\} \cdot t, t') & & (\hat{x}.(\{\vec{c} \mapsto \vec{u}\} \cdot t)x, t') & & (\{\vec{c} \mapsto \vec{u}\} \cdot t) t'
\end{array}$$

To express the rule CASELAM we need a morphism that abstracts the case construct *w.r.t.* a variable:

$$\begin{array}{ccc}
\mathbf{case}^\circ = \Lambda(f_{\mathbf{case}}) : & D^n \times D^D & \rightarrow D^D \\
& (\vec{u}, \hat{x}.t) & \mapsto \hat{x}. \{\vec{c} \mapsto \vec{u}\} \cdot t
\end{array}$$

where $f_{\mathbf{case}} = (D^n \times D^D) \times D \xrightarrow{\cong} D^n \times (D^D \times D) \xrightarrow{Id_{D^n} \times \mathbf{ev}} D^n \times D \xrightarrow{\mathbf{case}} D$.

Then the rule CASELAM is valid if (D4) commutes:

$$\begin{array}{ccccccc}
D^n \times D^D & \xrightarrow{\mathbf{case}^\circ} & D^D & \xrightarrow{\mathbf{lam}} & D & = & D^n \times D^D \xrightarrow{-\times \mathbf{lam}} D^n \times D \xrightarrow{\mathbf{case}} D \\
(\vec{u}, \hat{x}.t) & & \hat{x}. \{\vec{c} \mapsto \vec{u}\} \cdot t & & \lambda x. \{\vec{c} \mapsto \vec{u}\} \cdot t & & (\vec{u}, \hat{x}.t) \quad (\vec{u}, \lambda x.t) \quad \{\vec{c} \mapsto \vec{u}\} \cdot \lambda x.t
\end{array}$$

Also the rule CASECASE requires a morphism to compose case-bindings:

$$\begin{array}{ccc}
\bullet : & D^n \times D^n & \rightarrow D^n \\
& (\vec{u}, (t_i)_{i=1}^n) & \mapsto (\{\vec{c} \mapsto \vec{u}\} \cdot t_i)_{i=1}^n
\end{array}$$

It is defined as the pairing of the morphisms $(Id_{D^n} \times \pi_i^n); \mathbf{case}$, for $1 \leq i \leq n$. So it is the unique morphism that makes the diagram on the following commute.

$$\begin{array}{ccccc}
& & D^n \times D^n & & \\
& \swarrow Id \times \pi_1^n & & \searrow Id \times \pi_n^n & \\
D^n \times D & & \cdots & & D^n \times D \\
\downarrow \mathbf{case} & & \bullet & & \downarrow \mathbf{case} \\
D & & \cdots & & D \\
& \swarrow \pi_1^n & & \searrow \pi_n^n & \\
& & D^n & &
\end{array}$$

Then the commutation of the diagram (D5) validates the rule CASECASE.

This leads to the following definition.

Definition 2.1 ($\lambda_{\mathcal{C}}$ -model) A categorical model for the untyped $\lambda_{\mathcal{C}}$ -calculus is $\mathcal{M} = (\mathbb{C}, D, \mathbf{app}, \mathbf{lam}, (c_i^*)_{i=1}^n, \downarrow, \mathbf{case})$ where

- \mathbb{C} is a Cartesian closed category,
- D is an object of \mathbb{C} ,
- All the c_i^* 's and \downarrow are points of D ,
- \mathbf{app} is a morphism of $D \rightarrow D^D$, \mathbf{lam} is a morphism of $D^D \rightarrow D$ and \mathbf{case} a morphism of $D^n \times D \rightarrow D$,
- The six diagrams of Fig. 2 commute (the diagram (D2) must commute for every $i \in \llbracket 1..n \rrbracket$).

LAMAPP/APPLAM		CASECONS	
(D1)	$ \begin{array}{ccc} & Id_D & \\ D & \xrightarrow{\text{app}} & D^D \\ & \text{lam} & \\ & Id_{D^D} & \\ & \xrightarrow{\text{lam}} & D \end{array} $	(D2)	$ \begin{array}{ccc} D^n & \xrightarrow{\cong} & D^n \times \mathbf{1} \\ \pi_i^n \downarrow & & \downarrow Id \times c_i^* \\ D & \xleftarrow{\text{case}} & D^n \times D \end{array} $
CASEAPP		CASELAM	
(D3)	$ \begin{array}{ccc} (D^n \times D) \times D & \xrightarrow{\cong} & D^n \times (D \times D) \\ \text{case} \times Id \downarrow & & Id \times (\text{app} \times Id) \downarrow \\ D \times D & & D^n \times (D^D \times D) \\ \text{app} \times Id \downarrow & & Id \times ev \downarrow \\ D^D \times D & & D^n \times D \\ \searrow ev & & \swarrow \text{case} \\ & D & \end{array} $	(D4)	$ \begin{array}{ccc} D^n \times D^D & \xrightarrow{\text{case}^\circ} & D^D \\ Id \times lam \downarrow & & \downarrow lam \\ D^n \times D & \xrightarrow{\text{case}} & D \end{array} $
CASECASE			
(D5)	$ \begin{array}{ccc} (D^n \times D^n) \times D & \xrightarrow{\cong} & D^n \times (D^n \times D) \\ \bullet \times Id \downarrow & & Id \times \text{case} \downarrow \\ D^n \times D & & D^n \times D \\ \searrow \text{case} & & \swarrow \text{case} \\ & D & \end{array} $	(D6)	$ \begin{array}{ccc} D^n \times \mathbf{1} & \xrightarrow{Id_{D^n} \times \zeta} & D^n \times D \\ \pi_2 \downarrow & & \downarrow \text{case} \\ \mathbf{1} & \xrightarrow{\zeta} & D \end{array} $

Figure 2: Commuting diagrams in a $\lambda_{\mathcal{C}}$ -model

Equivalent definition. In fact we can simplify the definition of a $\lambda_{\mathcal{C}}$ -model, since the isomorphism $D \cong D^D$ entails the equivalence of the diagrams (D3) and (D4). This can be understood from a syntactical point of view, given that the commutation of the diagram (D3) validates the rule CASEAPP and the one of (D4) validates CASELAM. Indeed, the only role of CASELAM in the calculus is to close a critical pair created by the rule CASEAPP [1, Theo. 1, (CC3)].

Proposition 2.1 *If lam and app form an isomorphism between D and D^D , then the diagram (D3) commutes if and only if the diagram (D4) commutes.*

Proof:

Since (D1) commutes, (D4) commutes *iff* the diagram on the right commutes.

Write $f = Id_{D^n} \times lam; case; app$.

Since $case^\circ = \Lambda(\cong; Id_{D^n} \times ev; case)$, and by uniqueness of the exponential, $f = case^\circ$ if and only if the following diagram commutes:

$$\begin{array}{ccc}
(D^n \times D^D) \times D & \xrightarrow{\cong; Id_{D^n} \times ev; case} & D \\
f \times Id_D \downarrow & \nearrow ev & \\
D^D \times D & &
\end{array}$$

$$\begin{array}{ccc}
D^n \times D^D & \xrightarrow{case^\circ} & D^D \\
Id \times lam \downarrow & & \uparrow app \\
D^n \times D & \xrightarrow{case} & D
\end{array}$$

We can detail this diagram as follows:

$$\begin{array}{ccc}
(D^n \times D^D) \times D & \xrightarrow{\cong} & D^n \times (D^D \times D) \xrightarrow{Id_{D^n} \times ev} D^n \times D \\
(Id \times lam) \times Id \left(\cong \int \right) (Id \times app) \times Id \quad \curvearrowright & \uparrow Id_{D^n} \times (app \times Id_D) & \downarrow \text{case} \\
(D^n \times D) \times D & \xrightarrow{\cong} & D^n \times (D \times D) \\
\text{case} \times Id_D \downarrow & & (D3) \\
D \times D & \xrightarrow{app \times Id_D} & D^D \times D \xrightarrow{ev} D
\end{array}$$

Since the sub-diagram in the upper-left corner commutes, then (D4) commutes if and only if (D3) commutes. \square

Thus we can omit the commutation of (D3) or the one of (D4) in the definition of a $\lambda_{\mathcal{C}}$ -model.

2.2 Soundness

In the previous section we gave some intuitions on how to interpret $\lambda_{\mathcal{C}}$ -terms in a $\lambda_{\mathcal{C}}$ -model. Formally, the denotation $[t]_{\Gamma}$ of a term t in such a category is defined by structural induction (in Fig. 3). It depends on a list of variables $\Gamma = x_1, \dots, x_k$ that must contain all the free variables of t , and its a morphism of $D^k \rightarrow D$. Similarly, the denotation $[\theta]_{\Gamma}$ of a case-binding θ with free variables in Γ is a morphism of $D^k \rightarrow D^n$. We show that this definition provides a correct model of the $\lambda_{\mathcal{C}}$ -calculus (we write $\simeq_{\lambda_{\mathcal{C}}}$ for the reflexive symmetric transitive closure of its six rules).

$$\begin{aligned}
[x_i]_{\Gamma} &= \pi_i^k : D^k \rightarrow D \\
[tu]_{\Gamma} &= D^k \xrightarrow{\langle [t]_{\Gamma}; [u]_{\Gamma} \rangle} D \times D \xrightarrow{app \times Id_D} D^D \times D \xrightarrow{ev} D \\
[\lambda x_{k+1}.t]_{\Gamma} &= D^k \xrightarrow{\Lambda(f_t)} D^D \xrightarrow{lam} D \\
\text{where } f_t &= D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[t]_{\Gamma, x_{k+1}}} D \\
[c]_{\Gamma} &= D^k \xrightarrow{!_{D^k}} \mathbf{1} \xrightarrow{c^*} D \\
[\{\theta\} \cdot t]_{\Gamma} &= D^k \xrightarrow{\langle [\theta]_{\Gamma}; [t]_{\Gamma} \rangle} D^n \times D \xrightarrow{\text{case}} D \\
[\theta]_{\Gamma} &= \langle f_1; \dots; f_n \rangle : D^k \rightarrow D^n, \quad \text{where } f_i = \begin{cases} [u_i]_{\Gamma} & \text{if } c_i \mapsto u_i \in \theta \\ !_{D^k}; \downarrow & \text{if } c_i \notin \text{dom}(\theta) \end{cases}
\end{aligned}$$

Figure 3: Interpretation of $\lambda_{\mathcal{C}}$ -terms in a categorical model

Theorem 2.2 (Soundness) *If $\mathcal{M} = (\mathbb{C}, D, lam, app, (c_i^*)_{i=1}^n, case, \downarrow)$ is a $\lambda_{\mathcal{C}}$ -model, then for any $\lambda_{\mathcal{C}}$ -term t, t' whose free variables are in Γ ,*

$$t \simeq_{\lambda_{\mathcal{C}}} t' \implies [t]_{\Gamma} = [t']_{\Gamma}$$

To prove this theorem, we fix a $\lambda_{\mathcal{C}}$ -model $\mathcal{M} = (\mathbb{C}, D, lam, app, (c_i^*)_{i=1}^n, case, \downarrow)$ and use some preliminary lemmas. The first one expresses that the morphism \bullet actually corresponds to case-composition. This is where we technically need the diagram (D6), even though its semantic meaning is not as intuitive as for the other one.

Lemma 2.3 (Categorical case-composition) *If the diagram (D6) commutes, then for any case-bindings θ and ϕ , whose free variables are in $\Gamma = \{x_1, \dots, x_k\}$, the following diagram commute:*

$$\begin{array}{ccc} D^k & \xrightarrow{\langle [\theta]_\Gamma, [\phi]_\Gamma \rangle} & D^n \times D^n \\ & \searrow [\theta \circ \phi]_\Gamma & \downarrow \bullet \\ & & D^n \end{array}$$

Proof: If $\phi = \{c_i \mapsto u_i / i \in J\}$ (with $J \subseteq \llbracket 1..n \rrbracket$), then

$$[\theta \circ \phi]_\Gamma = \langle f_1, \dots, f_n \rangle, \quad \text{with } f_i = \begin{cases} [\{\theta\} \cdot u_i]_\Gamma & \text{if } i \in J \\ !_{D^k}; \downarrow & \text{if } i \notin J \end{cases}$$

On the other hand, $\bullet = \langle ((Id_{D^n} \times \pi_1^n); \text{case}), \dots, ((Id_{D^n} \times \pi_1^n); \text{case}) \rangle$. So

$$\langle [\theta]_\Gamma, [\phi]_\Gamma \rangle ; \bullet = \langle g_1, \dots, g_n \rangle, \quad \text{with } g_i = \langle [\theta]_\Gamma, ([\phi]_\Gamma ; \pi_i^n) \rangle ; \text{case}.$$

If $i \in J$, $[\phi]_\Gamma ; \pi_i^n = [u_i]_\Gamma$ and then $g_i = \langle [\theta]_\Gamma, [u_i]_\Gamma \rangle ; \text{case}$ which is f_i .

If $i \notin J$, then $[\phi]_\Gamma ; \pi_i^n = !_{D^k} \times \downarrow$. Hence

$$\begin{aligned} g_i &= D^k \xrightarrow{\langle [\theta]_\Gamma, !_{D^k} \rangle} D^n \times \mathbf{1} \xrightarrow{Id_{D^n} \times \downarrow} D^n \times D \xrightarrow{\text{case}} D \\ &= D^k \xrightarrow{\langle [\theta]_\Gamma, !_{D^k} \rangle} D^n \times \mathbf{1} \xrightarrow{\pi_2} \mathbf{1} \xrightarrow{\downarrow} D \quad (\text{by (D6)}) \\ &= D^k \xrightarrow{!_{D^k}} \mathbf{1} \xrightarrow{\downarrow} D \end{aligned}$$

So $g_i = f_i$ for any $i \leq n$, and $\langle [\theta]_\Gamma, [\phi]_\Gamma \rangle ; \bullet = [\theta \circ \phi]_\Gamma$. □

We also need the standard following lemmas.

Lemma 2.4 (Contextual rules) *Exchange: Let $\Gamma = \{x_1, \dots, x_k\}$ and σ be a substitution over $\llbracket 1..k \rrbracket$. Write $\sigma(\Gamma) = \{\sigma(1), \dots, \sigma(k)\}$. Then, for any term t whose free variables are in Γ ,*

$$[t]_\Gamma = \langle \pi_{\sigma(1)}^k, \dots, \pi_{\sigma(k)}^k \rangle ; [t]_{\sigma(\Gamma)}.$$

Weakening: Let $\Gamma = \{x_1, \dots, x_k\}$ containing all free variables of a term t , and $y \notin \Gamma$. Then

$$[t]_{\Gamma, y} = \langle \pi_1^{k+1}, \dots, \pi_k^{k+1} \rangle ; [t]_\Gamma.$$

Lemma 2.5 (Substitution) *Given $\Gamma = \{x_1, \dots, x_k\}$, and two terms t and u such that $\text{fv}(u) \subseteq \Gamma$ and $\text{fv}(t) \subseteq \Gamma \cup \{y\}$,*

$$[t[y := u]]_\Gamma = D^k \xrightarrow{\langle Id, [u]_\Gamma \rangle} D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[t]_{\Gamma, y}} D$$

The soundness theorem is then a direct corollary of the following proposition, that is proved (in appendix A) by structural induction:

Proposition 2.6 *If $\mathcal{M} = (\mathbb{C}, D, \text{lam}, \text{app}, (c_i^*)_{i=1}^n, \text{case}, \downarrow)$ is a $\lambda_{\mathcal{C}}$ model, then for any $\Gamma = \{x_1, \dots, x_k\}$ and any terms t_1, t_2 such that $\text{fv}(t_1) \subseteq \Gamma$ and $t_1 \rightarrow t_2$, the interpretation given in Fig. 3 satisfies $[t_1]_\Gamma = [t_2]_\Gamma$.*

3 Completeness

In this part we shall prove that the converse of Theo. 2.2 holds in absence of match failure. Namely if two terms have the same interpretation in any $\lambda_{\mathcal{C}}$ -model then they are convertible using the rules of the calculus. It means that, without match failure, the diagrams of Fig. 2 are minimal.

Theorem 3.1 (Completeness) *If t and t' are two hereditarily defined $\lambda_{\mathcal{C}}$ -terms such that in any categorical $\lambda_{\mathcal{C}}$ -model $[t]=[t']$, then*

$$t \simeq_{\lambda_{\mathcal{C}}} t'.$$

Notice that this theorem does not hold for undefined terms. Indeed, every match failure receives the same denotation \perp in any $\lambda_{\mathcal{C}}$ -model, even though they are not $\lambda_{\mathcal{C}}$ -convertible. The completeness result is established using the same method as [6]:

1. We define $\mathbb{P}ER_{\lambda_{\mathcal{C}}}$, the Cartesian closed category of partial equivalence relation compatible with $\simeq_{\lambda_{\mathcal{C}}}$.
2. In this syntactic category, we construct a $\lambda_{\mathcal{C}}$ -model \mathcal{M}_{synt} .
3. Then we show that if $[t] = [t']$ in \mathcal{M}_{synt} , then $t \simeq_{\lambda_{\mathcal{C}}} t'$.

3.1 Partial equivalence relations

Partial equivalence relations (PER) are commonly used to transform a model of the untyped lambda calculus into a model of the typed lambda-calculus [9, 18]. Yet we use them here to instantiate the definition of $\lambda_{\mathcal{C}}$ -models in the category of PER on $\lambda_{\mathcal{C}}$ -terms. Thereby we construct a syntactic model of the untyped $\lambda_{\mathcal{C}}$ -calculus.

Definition 3.1 ($\lambda_{\mathcal{C}}\text{-per}$) *Given a set X , a partial equivalence relation on X is a binary relation R that is symmetric and transitive. We may write $x = y : R$ instead of $(x, y) \in R$. A $\lambda_{\mathcal{C}}\text{-per}$ is a partial equivalence relation R on Λ (the set of all $\lambda_{\mathcal{C}}$ -terms) that is compatible with $\lambda_{\mathcal{C}}$ -equivalence, which means:*

$$\begin{cases} t = t' : R \\ t_0 \simeq_{\lambda_{\mathcal{C}}} t' \end{cases} \quad \text{implies} \quad t = t_0 : R$$

We write \bar{e}^R the equivalence class of an element e modulo R (or simply \bar{e} when it raises no ambiguity), and if it is non empty we say that e is *accessible* by R . This is denoted by $e \in R$. We call the *domain* of R (denoted by $\text{dom}(R)$) the set of all its accessible elements modulo R : $\text{dom}(R) = \{ \bar{e}^R / e \in R \}$. Notice that if a partial equivalence relation R is compatible with $\lambda_{\mathcal{C}}$ then by definition

$$t \simeq_{\lambda_{\mathcal{C}}} t' \quad \implies \quad \bar{t}^R = \bar{t}'^R. \quad (1)$$

It is well known that the family of partial equivalence relations can be provided with the usual semantic operators (arrow, and product) and constitute a CCC [15, Theo 7.1] To this end, we use the well-known Church's encoding for tuples:

$$\begin{aligned} \langle x_1, \dots, x_k \rangle_k &= \lambda f. f x_1 \dots x_k \\ \pi_i^k &= \lambda p. p (\lambda x_1 \dots x_k. x_i) \quad (i \in [1..k]) \end{aligned}$$

(We may write $\langle x, y \rangle$ for $\langle x, y \rangle_2$ and π_i for π_i^2). It satisfies the expected equivalence:

$$\pi_i^k \langle t_1, \dots, t_k \rangle_k \simeq_{\lambda_{\mathcal{C}}} t_i.$$

Proposition 3.2 (Operations on $\lambda_{\mathcal{C}}$ -pers) Let $(R_i)_{1 \leq i \leq n}$ be a family of PERs (with $n \geq 2$). Define $R_1 \rightarrow R_2$ and $R_1 \times \dots \times R_n$ by

$$\begin{aligned} t = t' : R \rightarrow R' & \quad \text{when} \quad \text{for any } u, u', u = u' : R \implies tu = t'u' : R' \\ t = u : R_1 \times \dots \times R_k & \quad \text{when} \quad \text{for each } i \in \llbracket 1..k \rrbracket, \quad \pi_i^k t = \pi_i^k u : R_i \end{aligned}$$

Then if all the R_i 's are $\lambda_{\mathcal{C}}$ -pers, so are $R_1 \rightarrow R_2$ and $R_1 \times \dots \times R_n$.

The category $\mathbb{P}er_{\lambda_{\mathcal{C}}}$. The previous proposition enables providing the category of $\lambda_{\mathcal{C}}$ -pers with the structure of a CCC. In the category $\mathbb{P}er_{\lambda_{\mathcal{C}}}$, objects are the PERs compatible with $\lambda_{\mathcal{C}}$, and given two $\lambda_{\mathcal{C}}$ -pers A and B the morphisms of $A \rightarrow B$ are the equivalence classes in $\text{dom}(A \rightarrow B)$. The identity morphism on A is $\overline{\lambda x.x}^{A \rightarrow A}$, and the composition of $\bar{t} : A \rightarrow B$ and $\bar{t}' : B \rightarrow C$ is $\bar{t}; \bar{t}' = \overline{\lambda z.t'(tz)}^{A \rightarrow C}$. This defines correctly a category, as the composition is associative and has identity morphisms as neutral elements.

The categorical product of two $\lambda_{\mathcal{C}}$ -pers A and B is $(A \times B, \overline{\pi_1}^{A \times B \rightarrow A}, \overline{\pi_2}^{A \times B \rightarrow B})$, and for $\bar{t} : C \rightarrow A$ and $\bar{t}' : C \rightarrow B$, the pairing of \bar{t}_1 and \bar{t}_2 is $\langle \bar{t}, \bar{t}' \rangle = \overline{\lambda x.(tx, t'x)}^{C \rightarrow A \times B}$. It is well defined (in particular it does not depend on the representative that we chose in the equivalence classes \bar{t} and \bar{t}') and is universal for the diagram on the right.

$$\begin{array}{ccccc} & & C & & \\ & \swarrow \bar{t} & \downarrow \langle \bar{t}, \bar{t}' \rangle & \searrow \bar{t}' & \\ A & \xleftarrow{\pi_1} & A \times B & \xrightarrow{\pi_2} & B \end{array}$$

The terminal object is the maximal $\lambda_{\mathcal{C}}$ -per $\mathbf{1} = \Lambda \times \Lambda$.

The exponent of A and B is $B^A = A \rightarrow B$, and the corresponding evaluation morphism is $\text{ev} = \overline{\lambda x.(\pi_1 x)(\pi_2 x)}^{B^A \times A \rightarrow B}$.

$$\begin{array}{ccc} C \times A & \xrightarrow{\bar{t}} & B \\ \Lambda(\bar{t}) \times \text{Id} \downarrow & \nearrow \text{ev} & \\ B^A \times A & & \end{array}$$

The curried form of a morphism $\bar{t} : C \times A \rightarrow B$ is then $\Lambda(\bar{t}) = \overline{\lambda x.\lambda y.t(x, y)}^{C \rightarrow B^A}$. It is well defined and is the unique morphism that makes the diagram on the left commute.

Proposition 3.3 $\mathbb{P}er_{\lambda_{\mathcal{C}}}$ is a Cartesian closed category.

3.2 Syntactic model in $\mathbb{P}er_{\lambda_{\mathcal{C}}}$.

We will now define a $\lambda_{\mathcal{C}}$ -model in the CCC $\mathbb{P}er_{\lambda_{\mathcal{C}}}$. In this category, there is a trivial reflexive object, that is actually equal to its object of functions (as proved in appendix B.1).

Lemma 3.4 Let D be the object $\simeq_{\lambda_{\mathcal{C}}}$ in $\mathbb{P}er_{\lambda_{\mathcal{C}}}$. Then $D = D^D$.

Also $\simeq_{\lambda_{\mathcal{C}}}$ is the object of $\mathbb{P}er_{\lambda_{\mathcal{C}}}$ that will be used to interpret untyped $\lambda_{\mathcal{C}}$ -terms. We do not need to define **lam** and **app**, and the morphisms c_i^* 's and **case** are quite intuitive: informally, c^* is the constant function returning **c**, and **case** takes an argument (θ, t) in $D^n \times D$ and return $\{\theta\} \cdot t$. In the same way, \downarrow is just a constant function returning a match failure (we arbitrarily choose one of the possible ones). This actually defines a $\lambda_{\mathcal{C}}$ -model (appendix B.1).

Definition 3.2 (Syntactic model) The syntactic model (or PER model) of the $\lambda_{\mathcal{C}}$ -calculus is $\mathcal{M}_{\text{synt}} = (\mathbb{P}er_{\lambda_{\mathcal{C}}}, D, \text{Id}_D, \text{Id}_D, (c_i^*)_{1 \leq i \leq n}, \text{case}, \downarrow)$, where:

- D is the relation $\simeq_{\lambda_{\mathcal{C}}}$.
- given **c** a constructor, c^* is $\overline{\lambda x.c}^{1^D}$.

- **case** is $\overline{\lambda x. \{ \{ c_i \mapsto \pi_i^n(\pi_1 x) \}_{1 \leq i \leq n} \} \cdot \pi_2 x}^{(D^n \times D) \rightarrow D}$.
- $\not\downarrow$ is $\overline{\lambda x. \{ \} \cdot c_1}^{1 \rightarrow D}$.

Proposition 3.5 \mathcal{M}_{synt} is a $\lambda_{\mathcal{C}}$ -model.

Case-binding completion. Remember that $\lambda_{\mathcal{C}}$ -models do not distinguish different match failures (as a matter of fact, all of them are interpreted by $\not\downarrow$). That is because the interpretation of a term first “completes” each case-binding with branches $c_j \mapsto \not\downarrow$ if c_j is not in its domain (cf. the description of the denotation of a case-binding page 3). Also in the PER model, undefined terms are “unblocked” and the rule CASECONS can be performed (and give $\{ \} \cdot c_1$). Now we formalise the idea of case-binding completion. This enables an explicit definition of the interpretation of a term in the PER model, so that we can prove the completeness theorem.

Definition 3.3 (Case-completion) The case-completion \tilde{t} of a term t is defined by induction:

$$\begin{aligned} \tilde{x} &= x & \widetilde{\lambda x. t} &= \lambda x. \tilde{t} & \widetilde{\{ \theta \} \cdot t} &= \{ \tilde{\theta} \} \cdot \tilde{t} \\ \tilde{c} &= c & \tilde{tu} &= \tilde{t}\tilde{u} \\ \tilde{\theta} &= \{ c_i \mapsto u'_i / 1 \leq i \leq n \} & \text{with } u'_i &= \begin{cases} \tilde{u}_i & \text{if } c_i \mapsto u_i \in \theta \\ \{ \} \cdot c_1 & \text{if } c_i \notin \text{dom}(\theta) \end{cases} \end{aligned}$$

Fact 3.4 This case-completion does not unify different defined terms: if two defined terms have the same case-completion, then they are equal.

Proposition 3.6 In the model \mathcal{M}_{synt} , the interpretation of a term t in a context $\Gamma = x_1; \dots; x_k$ is

$$[t]_{\Gamma} = \overline{\lambda x. \tilde{t}[x_i := \pi_i^k x]}^{D^k \rightarrow D} \quad (\text{with } x \text{ fresh in } t).$$

3.3 Completeness result.

The proposition 3.6 ensures that if two $\lambda_{\mathcal{C}}$ -terms have the same denotation in the PER model, then they have the same case-completion *modulo* D (i.e. they are $\lambda_{\mathcal{C}}$ -convertible). It does not necessarily means that the two terms are $\lambda_{\mathcal{C}}$ -equivalent themselves, as it is not true for match failure:

$$\begin{aligned} \{ c_1 \mapsto \widetilde{\lambda y. yy} \} \cdot c_2 &= \{ c_1 \mapsto \lambda y. yy; c_2 \mapsto \{ \} \cdot c_1 \} \cdot c_2 \simeq_{\lambda_{\mathcal{C}}} \{ \} \cdot c_1 \\ \{ c_2 \mapsto \lambda y. y \} \cdot c_1 &= \{ c_1 \mapsto \{ \} \cdot c_1; c_2 \mapsto \lambda y. y \} \cdot c_1 \simeq_{\lambda_{\mathcal{C}}} \{ \} \cdot c_1 \end{aligned}$$

Nevertheless, $\{ c_1 \mapsto \lambda y. yy \} \cdot c_2 \not\simeq_{\lambda_{\mathcal{C}}} \{ c_2 \mapsto \lambda y. y \} \cdot c_1$. This explains why match failure all have the same interpretation in \mathcal{M}_{synt} . However, this defect is restricted to undefined terms. Now we show that the case-completion does not modify the $\lambda_{\mathcal{C}}$ -equivalence on defined terms.

Proposition 3.7 Let t_1 and t_2 be two hereditarily defined terms. Then

$$\tilde{t}_1 \simeq_{\lambda_{\mathcal{C}}} \tilde{t}_2 \implies t_1 \simeq_{\lambda_{\mathcal{C}}} t_2$$

The proof of this proposition uses rewriting techniques, and relies on several lemmas (whose proofs are given in appendix B.2). For technical reasons, we need to separate the rule CASECASE from the other ones. Also we write $\lambda_{\mathcal{C}}^-$ the calculus with all the rules *except* CASECASE, and CC the rule CASECASE.

Fact 3.5 The definition of case-completion (Def. 3.3) preserves all $\lambda_{\mathcal{C}}$ -redexes. Also if $t \rightarrow u$ then $\tilde{t} \rightarrow \tilde{u}$, and if \tilde{t} is a normal form then so is t .

Lemma 3.8 (Reduction on completed terms) 1. Let t be a defined term.

Then, for any term t' ,

$$\tilde{t} \rightarrow_{\lambda_{\mathcal{C}}} t' \text{ implies } t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow t_0.$$

2. For any terms t, t' ,

$$\tilde{t} \rightarrow_{cc} t' \text{ implies } t' \rightarrow_{cc}^* \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow_{cc} t_0.$$

The rule CASECASE does not have the same behaviour as the other rules *w.r.t.* case-completion, and requires a special attention. It has been proved that the reduction rule CASECASE forms a confluent [1, Theo. 1] and strongly normalising [1, Prop. 2] rewriting system. So every $\lambda_{\mathcal{C}}$ -term t has a unique normal form $\Downarrow t$ for the rule CASECASE. It is characterised by the following equations:

$$\begin{array}{ll} \Downarrow x = x & \Downarrow \{c_i \mapsto u_i / i \in I\} = \{c_i \mapsto \Downarrow u_i / i \in I\} \\ \Downarrow c = c & \text{If } t = x \mid c \mid \lambda x. u \mid t_1 t_2, \text{ then} \\ \Downarrow \lambda x. t = \lambda x. \Downarrow t & \Downarrow \{\theta\} \cdot t = \Downarrow \{\theta\} \cdot \Downarrow t \\ \Downarrow (tu) = \Downarrow t \Downarrow u & \Downarrow (\{\theta\} \cdot \{\phi\} \cdot t) = \Downarrow (\{\theta \circ \phi\} \cdot t) \end{array}$$

Lemma 3.9 *Commutation case-completion/CC-normal form*

For any term t ,

$$\Downarrow (\tilde{t}) = \widetilde{\Downarrow t}.$$

Lemma 3.10 For any terms t, t' , if $t \rightarrow_{\lambda_{\mathcal{C}}} t'$ then there exists a term u such that

$$\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}}^* u \rightarrow_{cc}^* \Downarrow t'.$$

Corollary 3.11 If t is hereditarily defined, then for any t' ,

$$\tilde{t} \rightarrow^* t' \text{ implies } \Downarrow t' = \tilde{t}_0 \text{ for some } t_0 \text{ such that } t \rightarrow^* t_0.$$

Proof: By induction on the reduction $\tilde{t} \rightarrow^* t'$.

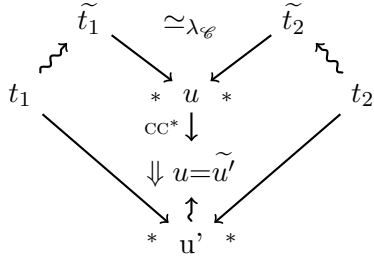
If $\tilde{t} = t'$, take $t_0 = \Downarrow t$. Now assume $\tilde{t} \rightarrow^* u \rightarrow_R t'$. By induction hypothesis, there is some u_0 such that $\Downarrow u = \widetilde{u_0}$ and $t \rightarrow^* u_0$. If u reduces on t' with the rule $R = \text{CASECASE}$, then $\Downarrow t' = \Downarrow u = \widetilde{u_0}$, and $t_0 = u_0$ does the job. Otherwise, $\tilde{t} \xrightarrow{\lambda_{\mathcal{C}}}^* u \rightarrow_{\lambda_{\mathcal{C}}} t'$.

$$\begin{array}{ccccc} \tilde{t} & \xrightarrow{\quad}^* & u & \xrightarrow{\quad}^* & t' \\ \uparrow \text{wavy} & & \vdots \text{CC}^* & & \vdots \text{CC}^* \\ & & \Downarrow u = \widetilde{u_0} \xrightarrow{\lambda_{\mathcal{C}}}^* \widetilde{u_1} \xrightarrow{\text{CC}^*} \Downarrow t' = \widetilde{u_1} & & \\ & & \updownarrow & & \updownarrow \\ t & \xrightarrow{\quad}^* & u_0 & \xrightarrow{\quad}^* & u_1 \xrightarrow{\text{CC}^*} \Downarrow u_1 \end{array}$$

First of all, $u \rightarrow_{\lambda_{\mathcal{C}}} t'$ implies $\Downarrow u \rightarrow_{\lambda_{\mathcal{C}}}^* u' \rightarrow_{cc}^* \Downarrow t'$ for some u' (Lem. 3.10). Also $\widetilde{u_0} \rightarrow_{\lambda_{\mathcal{C}}}^* u'$, and thus $u' = \widetilde{u_1}$ for some term u_1 such that $u_0 \rightarrow_{\lambda_{\mathcal{C}}}^* u_1$ (Lem. 3.8.1, since u_0 is defined). Moreover, $\widetilde{u_1} \rightarrow_{cc}^* \Downarrow t'$ implies that $\Downarrow t'$ is the CASECASE normal form of $\widetilde{u_1}$. Hence $\Downarrow t' = \Downarrow \widetilde{u_1} = \Downarrow u_1$ (by Lem. 3.9). Also we can chose $t_0 = \Downarrow u_1$. \square

Now we have all the ingredients we need to prove that the case-completion preserves the $\lambda_{\mathcal{C}}$ -equivalence on hereditarily defined terms.

Proof: (of Prop. 3.7).



Let t_1, t_2 hereditarily defined such that $\tilde{t}_1 \simeq_{\lambda_{\mathcal{C}}} \tilde{t}_2$. Since the $\lambda_{\mathcal{C}}$ -calculus satisfies the Church-Rösser property, there is a term u such that $\tilde{t}_1 \rightarrow^* u$ and $\tilde{t}_2 \rightarrow^* u$.

Hence Cor. 3.11 provides a term u' such that $\Downarrow u = u'$, and $t_i \rightarrow^* u'$ for each $i \in \{1, 2\}$. Thus $t_1 \simeq_{\lambda_{\mathcal{C}}} u' \simeq_{\lambda_{\mathcal{C}}} t_2$. \square

Together with the explicit definition of the interpretation of a term in the PER-model, this gives the result of completeness of $\lambda_{\mathcal{C}}$ -models for terms with no match failure.

Corollary 3.12 (Completeness) *Let t_1 and t_2 be two hereditarily defined terms whose free variables are in $\Gamma = \{x_1, \dots, x_k\}$ such that $[t_1]_{\Gamma} = [t_2]_{\Gamma}$ in the syntactic model \mathcal{M}_{synt} , then $t_1 \simeq_{\lambda_{\mathcal{C}}} t_2$.*

Proof: By Prop. 3.6, if t_1 and t_2 have the same interpretation in \mathcal{M}_{synt} , it means that

$$\overline{\lambda x. \tilde{t}_1[x_i := \pi_i^k x]}^{D^k \rightarrow D} = \overline{\lambda x. \tilde{t}_2[x_i := \pi_i^k x]}^{D^k \rightarrow D}.$$

Hence $(\lambda x. \tilde{t}_1[x_i := \pi_i^k x]) \langle x_1, \dots, x_k \rangle_k = (\lambda x. \tilde{t}_2[x_i := \pi_i^k x]) \langle x_1, \dots, x_k \rangle_k : D$. Since D is the $\lambda_{\mathcal{C}}$ -equivalence relation on terms, it means that $\tilde{t}_1 \simeq_{\lambda_{\mathcal{C}}} \tilde{t}_2$, which entails $t_1 \simeq_{\lambda_{\mathcal{C}}} t_2$ by Prop. 3.7. \square

A fortiori if two hereditarily defined terms have the same interpretation in *any* $\lambda_{\mathcal{C}}$ -model then they are $\lambda_{\mathcal{C}}$ -equivalent, since \mathcal{M}_{synt} is a $\lambda_{\mathcal{C}}$ -model. This achieves the proof of Completeness theorem (Theo. 3.1).

Notice that the separation theorem for the lambda calculus with constructors [1, Theo. 2] specifies that two hereditarily defined terms are either $\lambda_{\mathcal{C}}$ -equivalent or (weakly) separable. So any terms that can be separated by this syntactic lemma are also semantically distinguished by our definition of model. However a slight modification of this definition could allow to semantically separate more terms. If, instead of having one fail constant ζ we had one for each constructor (say $\zeta_1, fail_2$ etc.), we could “complete” a case binding with the corresponding fail constant in each undefined branch. This would enable keeping track of the constructor that raises the match failure. For instance, $\{c_1 \mapsto \lambda x.x\} \cdot c_2$ would be denoted by ζ_2 and $\{c_1 \mapsto \lambda x.x\} \cdot c_3$ by ζ_3 . Only terms like $\{c_1 \mapsto \lambda x.x\} \cdot c_2$ and $\{c_3 \mapsto \lambda x.x\} \cdot c_2$ would not be semantically separated.

Conclusion

We have defined a notion of categorical model for the lambda calculus with constructors that is reasonably complex: in addition to the usual axioms of a CCC, it involves three morphisms (or family of morphisms) and the commutation of six simple diagrams. We have also proved that this categorical model is complete for terms with no match failure.

Still, completeness does not hold for match failures. This is due to the way we interpret the case-bindings. Since the denotation we give to them is a point of D^n , it requires to “fill” artificially every undefined branch of a case-binding. A way to cope with this problem could be to first identify the domain $I \subseteq \llbracket 1..n \rrbracket$ of a case-binding $\theta = \{c_i \mapsto u_i / i \in I\}$, and interpret it by the point $(u_i)_{i \in I}$ of D^{n_I} (where n_I is the cardinal of I). The object that represents case-bindings

would then be the sum (the dual notion of product) $\sum_{I \subseteq \llbracket 1..n \rrbracket} D^{n_I}$. However, the definition loses its relative simplicity and some difficulties arise to define the case composition.

Future work A natural question is now to find some concrete instances of the categorical model. The PER model is one, but it would be of great interest to have some non syntactic models. We could try to adapt the historically first model of the pure lambda calculus [14]. However there is no reason for the usual Scott’s D_∞ domain to satisfy the commutation of our diagrams. A first step could be to find out a domain equation to characterise the lambda calculus with constructors, and then solve it with Scott’s technique.

An other issue is to define a categorical model for the *typed* $\lambda_{\mathcal{C}}$ -calculus [13]. This type system is rather complex, basically because of the reduction rule CASEAPP that transforms a sub-term that is *a priori* a function into a sub-term that is *a priori* a data-structure. To deal with this difficulty (and also to enable the typing of variadic constructors), the type syntax includes an application construct and the type system uses sub-typing. Also defining a typed categorical model for the lambda calculus with constructors probably requires a categorical definition of this type application, and a way to express categorically this sub-typing relation.

References

- [1] Ariel Arbiser, Alexandre Miquel, and Alejandro Ríos. A lambda-calculus with constructors. In *RTA*, pages 181–196, 2006.
- [2] Ariel Arbiser, Alexandre Miquel, and Alejandro Ríos. The lambda-calculus with constructors: Syntax, confluence and separation. *Journal of Functional Programming*, 19(5):581–631, 2009.
- [3] Andrea Asperti and Giuseppe Longo. *Categories, Types and Structures*. M.I.T. Press, 1991.
- [4] Horatiu Cirstea and Claude Kirchner. The rewriting calculus as a semantics of elan. In *ASIAN*, pages 84–85, 1998.
- [5] Thierry Coquand. Pattern matching with dependent types. In *Workshop on Types for Proofs and Programs*, pages 66–79, 1992.
- [6] Germain Faure and Alexandre Miquel. A categorical semantics for the parallel lambda-calculus. Technical Report 7063, INRIA, October 2009.
- [7] Martin Hofmann and Thomas Streicher. Continuation models are universal for lambda-mu-calculus. In *LICS*, pages 387–395, 1997.
- [8] C. Barry Jay and Delia Kesner. Pure pattern calculus. In *ESOP*, pages 100–114, 2006.
- [9] John C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions (summary). In *LISP and Functional Programming*, pages 308–319, 1986.
- [10] John C. Mitchell. *Foundations for Programming Languages*. The MIT Press, 1996.
- [11] C.-H. Luke Ong and Steven James Ramsay. Verifying higher-order functional programs with pattern-matching algebraic data types. In *Principles of Programming Languages*, pages 587–598, 2011.

- [12] Barbara Petit. *On the lambda calculus with constructors*. PhD thesis, ENS Lyon, 2011.
- [13] Barbara Petit. Semantics of typed lambda-calculus with constructors. *Logical Methods in Computer Science*, 7(1:2), 2011.
- [14] Dana S. Scott. Outline of a mathematical theory of computation. Technical report, Princeton University, 1970.
- [15] Dana S. Scott. Data types as lattices. *Siam Journal on Computing*, 5(3):522–587, 1976.
- [16] Dana S. Scott. Relating theories of the lambda-calculus. In Roger Hindley and Jonathan P. Seldin, editors, *To H.B. Curry: essays in Combinarory Logic, lambda calculus and Formalisms*. Academic Press, 1980.
- [17] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.
- [18] Val Tannen and Thierry Coquand. Extensional models for polymorphism. In *TAPSOFT, Vol.2*, pages 291–307, 1987.
- [19] Vincent van Oostrom. Lambda calculus with patterns. Technical report, Vrije Universiteit, Amsterdam, 1990.

A Proof of Soundness

Proposition 2.6. *If $\mathcal{M} = (\mathbb{C}, D, \text{lam}, \text{app}, (c_i^*)_{i=1}^n, \text{case}, \dagger)$ is a $\lambda_{\mathcal{C}}$ -model, then for any $\Gamma = \{x_1, \dots, x_k\}$ and any terms t_1, t_2 such that $\text{fv}(t_1) \subseteq \Gamma$ and $t_1 \rightarrow t_2$, the interpretation given in Fig. 3 satisfies $[t_1]_{\Gamma} = [t_2]_{\Gamma}$.*

Proof: Let t_1, t_2 be two $\lambda_{\mathcal{C}}$ -terms such that $t_1 \rightarrow t_2$. We prove by induction on the structure of t_1 that for any Γ containing all free variables of t_1 , $[t_1]_{\Gamma} = [t_2]_{\Gamma}$. If the reduction does not involve a head redex, we immediately conclude with induction hypothesis. So we consider all possible reductions in head position:

- $t_1 = (\lambda x.t) u$ and $t_2 = t[x := u]$.

$$[t_1]_{\Gamma} = D^k \xrightarrow{\langle (\Lambda(f_t); \text{lam}), [u]_{\Gamma} \rangle} D \times D \xrightarrow{\text{app} \times Id_D} D^D \times D \xrightarrow{\text{ev}} D$$

with $f_t = D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[t]_{\Gamma, x}} D$. Thus

$$\begin{aligned} [t_1]_{\Gamma} &= \langle Id_D, [u]_{\Gamma} \rangle ; (\Lambda(f_t); \text{lam}; \text{app}) \times Id_D ; \text{ev} \\ &= \langle Id_D, [u]_{\Gamma} \rangle ; \Lambda(f_t) \times Id_D ; \text{ev} & (D1) \\ &= \langle Id_D, [u]_{\Gamma} \rangle ; f_t & (\text{Def. of exponential}) \\ &= [t[x := u]]_{\Gamma} & (\text{Lem. 2.5}) \end{aligned}$$

- $t_1 = \lambda x.tx$ (with $x \notin \text{fv}(t)$) and $t_2 = t$. Then $[t_1]_{\Gamma} = \Lambda(f_{tx}) ; \text{lam}$

$$\text{where } f_{tx} = D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[t]_{\Gamma, x}, [x]_{\Gamma, x}} D \times D \xrightarrow{\text{app} \times Id_D} D^D \times D \xrightarrow{\text{ev}} D.$$

But $x \notin \text{fv}(t)$ implies $[t]_{\Gamma, x} = \langle \pi_1^{k+1}, \dots, \pi_k^{k+1} \rangle$; $[t]_{\Gamma}$ by weakening property (Lem. 2.4), and $[x]_{\Gamma, x} = \pi_{k+1}^{k+1}$.

$$\text{So } f_{tx} = D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{\langle \pi_1^{k+1}, \dots, \pi_k^{k+1} \rangle, \pi_{k+1}^{k+1}} D^k \times D \xrightarrow{([t]_{\Gamma}; \text{app}) \times Id_D} D^D \times D \xrightarrow{\text{ev}} D.$$

$\xrightarrow{Id_{D^k \times D}}$

By uniqueness of the exponential, $\Lambda(f_{tx}) = [t]_{\Gamma}; \text{app}$, and $[t_1]_{\Gamma} = [t]_{\Gamma}; \text{app}; \text{lam} = [t]_{\Gamma}$ by (D1).

- $t_1 = \{\theta\} \cdot c_i$ and $t_2 = u_i$, where $\theta = \{c_j \mapsto u_j / j \in J\}$, with $J \subseteq [1..n]$.

Then $[t_1]_{\Gamma} = \langle \langle f_1, \dots, f_n \rangle, [c_i]_{\Gamma} \rangle ; \text{case}$ with $f_j = \begin{cases} [u_j]_{\Gamma} & \text{if } j \in J \\ !_{D^k}; \dagger & \text{otherwise} \end{cases}$

and $[c_i]_{\Gamma} = !_{D^k}; c_i^*$.

The following diagram commutes:

$$\begin{array}{ccccc} D^k & \xrightarrow{\langle \langle f_1, \dots, f_n \rangle, !_{D^k} \rangle} & D^n \times \mathbf{1} & \xrightarrow{Id_{D^n} \times c_i^*} & D^n \times D \\ & \searrow \langle f_1, \dots, f_n \rangle & \cong \downarrow & (D2) & \downarrow \text{case} \\ & & D^n & \xrightarrow{\pi_i^n} & D \end{array}$$

so $[t_1]_{\Gamma} = \langle f_1, \dots, f_n \rangle ; \pi_i^n = f_i = [u_i]_{\Gamma}$.

- $t_1 = \{\theta\} \cdot (tu)$ and $t_2 = (\{\theta\} \cdot t) u$.

$$[t_1]_{\Gamma} = \langle [\theta]_{\Gamma}, [tu]_{\Gamma} \rangle ; \text{case} \text{ with } [tu]_{\Gamma} = \langle [t]_{\Gamma}, [u]_{\Gamma} \rangle ; (\text{app} \times Id_D) ; \text{ev}$$

$$[t_2]_{\Gamma} = \langle (\langle [\theta]_{\Gamma}, [t]_{\Gamma} \rangle ; \text{case}), [u]_{\Gamma} \rangle ; (\text{app} \times Id_D) ; \text{ev}$$

So $[t_1]_{\Gamma} = [t_2]_{\Gamma}$ because the following diagram commutes:

$$\begin{array}{ccccc}
& & D^n \times (D \times D) & \xrightarrow{Id \times (app \times Id)} & D^n \times (D^D \times D) & \xrightarrow{Id \times ev} & D^n \times D & \xrightarrow{\text{case}} & D \\
\langle [\theta]_\Gamma, \langle [t]_\Gamma, [u]_\Gamma \rangle \rangle & \nearrow & \uparrow \cong & & (D3) & & & & \\
D^k & & & & & & & & \\
\langle \langle [\theta]_\Gamma, \langle [t]_\Gamma \rangle \rangle, [u]_\Gamma \rangle & \searrow & \downarrow \cong & & & & & & \\
& & (D^n \times D) \times D & \xrightarrow{\text{case} \times Id} & D \times D & \xrightarrow{app \times Id} & D^D \times D & \xrightarrow{ev} & D
\end{array}$$

- $t_1 = \{\theta\} \cdot \lambda x. t$ and $t_2 = \lambda x. \{\theta\} \cdot t$ with $x \notin \text{fv}(\theta)$.

$$\begin{aligned}
[t_1]_\Gamma &= \langle [\theta]_\Gamma, (\Lambda(f_t); \mathbf{lam}) \rangle ; \mathbf{case} \quad \text{with } f_t = D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{[t]_\Gamma, x} D, \text{ and} \\
[t_2]_\Gamma &= \Lambda(f_{\{\theta\} \cdot t}); \mathbf{lam} \quad \text{with } f_{\{\theta\} \cdot t} = D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{\langle [\theta]_\Gamma, x, [t]_\Gamma, x \rangle} D^n \times D \xrightarrow{\text{case}} D. \\
\text{So } [t_1]_\Gamma &= \langle [\theta]_\Gamma, (\Lambda(f_t); \mathbf{lam}) \rangle ; \mathbf{case} \\
&= \langle [\theta]_\Gamma, \Lambda(f_t) \rangle ; (Id_{D^n} \times \mathbf{lam}) ; \mathbf{case} \\
&= \langle [\theta]_\Gamma, \Lambda(f_t) \rangle ; \mathbf{case}^\circ ; \mathbf{lam} \quad \text{by (D4)}
\end{aligned}$$

Hence $[t_1]_\Gamma = [t_2]_\Gamma$ if $\langle [\theta]_\Gamma, \Lambda(f_t) \rangle ; \mathbf{case}^\circ = \Lambda(f_{\{\theta\} \cdot t})$.

Remember that $\mathbf{case}^\circ = \Lambda(f_{\mathbf{case}})$, with

$$f_{\mathbf{case}} = (D^n \times D^D) \times D \xrightarrow{\cong} D^n \times (D^D \times D) \xrightarrow{Id_{D^n} \times ev} D^n \times D \xrightarrow{\mathbf{case}} D. \quad \text{To simplify this equation, we use this intermediate lemma (that follows from the uniqueness of exponent).}$$

Lemma A.1 *In any CCC, given four objects A, B, C and C' , and three morphisms $g : C \times A \rightarrow B$, $g' : C' \times A \rightarrow B$ and $h : C \rightarrow C'$,*

$$\Lambda(g) = h; \Lambda(g') \iff g = (h \times Id_A); g'.$$

Thus $[t_1]_\Gamma = [t_2]_\Gamma$ if $(\langle [\theta]_\Gamma, \Lambda(f_t) \rangle \times Id_d) ; f_{\mathbf{case}} = f_{\{\theta\} \cdot t}$.

Remark that $(\langle [\theta]_\Gamma, \Lambda(f_t) \rangle \times Id_d) ; f_{\mathbf{case}} = \text{lhs} ; \mathbf{case}$, with

$$\begin{aligned}
\text{lhs} &= D^k \times D \xrightarrow{\langle [\theta]_\Gamma, \Lambda(f_t) \rangle \times Id_D} (D^n \times D^D) \times D \xrightarrow{\cong} D^n \times (D^D \times D) \xrightarrow{Id_{D^n} \times ev} D^n \times D \\
&= D^k \times D \xrightarrow{\langle (\pi_1; [\theta]_\Gamma), Id \rangle} D^n \times (D^k \times D) \xrightarrow{Id_{D^n} \times (\Lambda(f_t) \times Id_D)} D^n \times (D^D \times D) \xrightarrow{Id_{D^n} \times ev} D^n \times D \\
&= D^k \times D \xrightarrow{\langle (\pi_1; [\theta]_\Gamma), Id \rangle} D^n \times (D^k \times D) \xrightarrow{Id_{D^n} \times f_t} D^n \times D
\end{aligned}$$

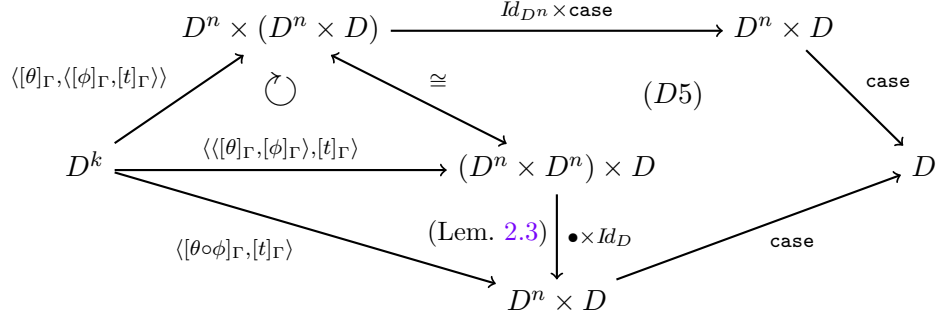
On the other hand, $f_{\{\theta\} \cdot t} = \text{rhs} ; \mathbf{case}$, with

$$\begin{aligned}
\text{rhs} &= D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{\langle Id, Id \rangle} D^{k+1} \times D^{k+1} \xrightarrow{[\theta]_\Gamma, x \times [t]_\Gamma, x} D^n \times D \\
&= D^k \times D \xrightarrow{\cong} D^{k+1} \xrightarrow{\langle Id, Id \rangle} D^{k+1} \times D^{k+1} \xrightarrow{\langle \dots, \pi_k^{k+1} \rangle \times Id} D^k \times D^{k+1} \xrightarrow{[\theta]_\Gamma \times [t]_\Gamma, x} D^n \times D \quad (\text{Lem. 2.4}) \\
&= D^k \times D \xrightarrow{\langle Id, Id \rangle} (D^k \times D) \times (D^k \times D) \xrightarrow{\pi_1 \times \cong} D^k \times D^{k+1} \xrightarrow{[\theta]_\Gamma \times [t]_\Gamma, x} D^n \times D \\
&= D^k \times D \xrightarrow{\langle Id, Id \rangle} (D^k \times D) \times (D^k \times D) \xrightarrow{(\pi_1; [\theta]_\Gamma) \times f_t} D^n \times D
\end{aligned}$$

Finally $\text{rhs} = \text{lhs} = \langle (\pi_1; [\theta]_\Gamma), f_t \rangle$, and so $[t_1]_\Gamma = [t_2]_\Gamma$.

- $t_1 = \{\theta\} \cdot \{\phi\} \cdot t$ and $t_2 = \{\theta \circ \phi\} \cdot t$.
 $[t_1]_\Gamma = (\langle [\theta]_\Gamma, \langle [\phi]_\Gamma, [t]_\Gamma \rangle \rangle) ; (Id_{D^n} \times \mathbf{case}) ; \mathbf{case}$, and
 $[t_2]_\Gamma = (\langle [\theta \circ \phi]_\Gamma, [t]_\Gamma \rangle) ; \mathbf{case}$.

Both terms have the same interpretation if the following diagram commute:



The upper triangle commutes by uniqueness of the product, the triangle below commutes if (D6) commutes (consequence of Lem. 2.3), and the right part of the diagram is exactly (D5). Also the interpretation is correct *w.r.t.* CASECASE if (D5) and (D6) commute. \square

B Proofs for Completeness

B.1 Some properties of $\mathbb{P}er_{\lambda_{\mathcal{C}}}$.

Lemma 3.4. *Let D be the object $\simeq_{\lambda_{\mathcal{C}}}$ in $\mathbb{P}er_{\lambda_{\mathcal{C}}}$. Then $D = D^D$.*

Proof:

\subseteq : If $t = t' : D$, then $u = u' : D$ implies $tu = t'u' : D$ by definition of D . This means $t = t' : D^D$

\supseteq : Assume $t = t' : D^D$, and choose x not free in t nor t' . Since $x = x : D$, then $tx = t'x : D$.

So $\lambda x.tx = \lambda x.t'x : D$ by contextual closure, and $t = t' : D$ by LAMAPP. \square

Proposition 3.5. *Let $\mathcal{M}_{\text{synt}} = (\mathbb{P}er_{\lambda_{\mathcal{C}}}, D, Id_D, Id_D, (c_i^*)_{1 \leq i \leq n}, \text{case}, \downarrow)$, where:*

- D is the relation $\simeq_{\lambda_{\mathcal{C}}}$.
- given c a constructor, c^* is $\overline{\lambda x.c} \mathbf{1}^D$.
- case is $\overline{\lambda x. \{(c_i \mapsto \pi_i^n(\pi_1 x))_{1 \leq i \leq n}\} \cdot \pi_2 x}^{(D^n \times D) \rightarrow D}$.
- \downarrow is $\overline{\lambda x. \{\} \cdot c_1} \mathbf{1}^{D \rightarrow D}$.

$\mathcal{M}_{\text{synt}}$ is a $\lambda_{\mathcal{C}}$ -model.

Proof: $\mathbb{P}er_{\lambda_{\mathcal{C}}}$ is a Cartesian closed category by Prop. 3.3, and Id_D is an isomorphism from D to D^D by Lem. 3.4. We first check that the morphisms are well-defined:

- $c^* \in \text{dom}(\mathbf{1} \rightarrow D)$ for each constructor c . Indeed, for any terms u, u' , $(\lambda x.c) u \simeq_{\lambda_{\mathcal{C}}} c \simeq_{\lambda_{\mathcal{C}}} (\lambda x.c) u'$. Hence $\lambda x.c = \lambda x.c : \mathbf{1} \rightarrow D$. In the same way, $\downarrow \in \text{dom}(\mathbf{1} \rightarrow D)$.
- $\text{case} \in \text{dom}(D^n \times D \rightarrow D)$ since $\lambda x. \{(c_i \mapsto \pi_i^n(\pi_1 x))_{i=1}^n\} \cdot \pi_2 x \in (D^n \times D) \rightarrow D$. Indeed, let $t = u : (D^n \times D)$. By definition, $\pi_i^n(\pi_1 t) = \pi_i^n(\pi_1 u) : D$, and $\pi_2 t = \pi_2 u : D$. Thus

$$\begin{aligned}
 (\lambda x. \{(c_i \mapsto \pi_i^n(\pi_1 x))_{i=1}^n\} \cdot \pi_2 x) t &\simeq_{\lambda_{\mathcal{C}}} \{(c_i \mapsto \pi_i^n(\pi_1 t))_{i=1}^n\} \cdot \pi_2 t \\
 &\simeq_{\lambda_{\mathcal{C}}} \{(c_i \mapsto \pi_i^n(\pi_1 u))_{i=1}^n\} \cdot \pi_2 u \\
 &\simeq_{\lambda_{\mathcal{C}}} (\lambda x. \{(c_i \mapsto \pi_i^n(\pi_1 x))_{i=1}^n\} \cdot \pi_2 x) u
 \end{aligned}$$

Finally by Prop. 2.1 it is sufficient to show that the diagrams (D1), (D2), (D3), (D5) and (D6) of Fig. 2 commute. For (D1) it is obvious with $\text{lam} = \text{app} = Id_D$. We show the commutation property for the other diagram.

(D2): We show that $rhs = \pi_i^n$, where $rhs = h_{\cong} ; (Id_{D^n} \times c_i^*) ; \text{case}$ (with $h_{\cong} = \overline{\lambda x. \langle x, x \rangle}^{D^n \rightarrow D^n \times 1}$). Notice that $(Id_{D^n} \times c_i^*) = \overline{\lambda x. \langle \pi_1 x, (\lambda x. c_i)(\pi_2 x) \rangle}$. We simplify rhs, considering terms up to $\lambda_{\mathcal{C}}$ -equivalence (1).

$$\begin{aligned}
rhs &= \overline{\lambda z. t_{\text{case}}((\lambda x. \langle \pi_1 x, (\lambda x. c_i) x \rangle) ((\lambda x. \langle x, x \rangle) z))}^{D^n \rightarrow D} \\
&= \overline{\lambda z. t_{\text{case}}(\langle \pi_1 \langle z, z \rangle, (\lambda x. c_i)(\pi_2 \langle z, z \rangle) \rangle)}^{D^n \rightarrow D} \\
&= \overline{\lambda z. t_{\text{case}}(\langle z, c_i \rangle)}^{D^n \rightarrow D} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n(\pi_1 \langle z, c_i \rangle)\}_{i=1}^n \cdot \pi_2 \langle z, c_i \rangle \rangle}^{D^n \rightarrow D} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n(\pi_1 \langle z, c_i \rangle)\}_{i=1}^n \cdot \pi_2 \langle z, c_i \rangle \rangle}^{D^n \rightarrow D} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n z\}_{i=1}^n \cdot c_i \rangle}^{D^n \rightarrow D} \\
&= \overline{\lambda z. \pi_i^n z}^{D^n \rightarrow D} \quad \text{by CASECONS} \\
&= \pi_i^n
\end{aligned}$$

(D3): We show that $lhs = rhs$, where $lhs = (\text{case} \times Id_D) ; (\text{app} \times Id_D) ; \text{ev}$, and $rhs = h_{\cong} ; (Id_{D^n} \times (\text{app} \times Id_D)) ; (Id_{D^n} \times \text{ev}) ; \text{case}$, with

$$h_{\cong} = \overline{\lambda x. \langle \pi_1(\pi_1 x), \langle \pi_2(\pi_1 x), \pi_2 x \rangle \rangle}^{(D^n \times D) \times D \rightarrow D^n \times (D \times D)}.$$

Notice that $\text{app} \times Id_D = Id_{D \times D}$, so $lhs = (\text{case} \times Id_D) ; \text{ev}$, and

$$rhs = h_{\cong} ; (Id_{D^n} \times \text{ev}) ; \text{case}.$$

$$\begin{aligned}
lhs &= \overline{\lambda z. (\lambda x. (\pi_1 x)(\pi_2 x)) ((\lambda x. \langle t_{\text{case}}(\pi_1 x), \pi_2 x \rangle) z)} \\
&= \overline{\lambda z. (\lambda x. (\pi_1 x)(\pi_2 x)) (\langle t_{\text{case}}(\pi_1 z), \pi_2 z \rangle)} \\
&= \overline{\lambda z. (t_{\text{case}}(\pi_1 z)) (\pi_2 z)} \\
&= \overline{\lambda z. (\langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z))\}_{i=1}^n \cdot \pi_2(\pi_1 z) \rangle) (\pi_2 z)} \\
rhs &= \overline{\lambda z. t_{\text{case}} (\lambda y. \langle \pi_1 y, (\lambda x. (\pi_1 x)(\pi_2 x))(\pi_2 y) \rangle) ((\lambda x. \langle \pi_1(\pi_1 x), \langle \pi_2(\pi_1 x), \pi_2 x \rangle \rangle) z)} \\
&= \overline{\lambda z. t_{\text{case}} (\lambda y. \langle \pi_1 y, (\pi_1(\pi_2 y))(\pi_2(\pi_2 y)) \rangle) (\langle \pi_1(\pi_1 z), \langle \pi_2(\pi_1 z), \pi_2 z \rangle \rangle)} \\
&= \overline{\lambda z. t_{\text{case}} (\langle \pi_1(\pi_1 z), (\pi_2(\pi_1 z))(\pi_2 z) \rangle)} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z))\}_{i=1}^n \cdot (\pi_2(\pi_1 z) (\pi_2 z)) \rangle} \\
&= \overline{\lambda z. (\langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z))\}_{i=1}^n \cdot \pi_2(\pi_1 z) \rangle) (\pi_2 z)} \quad \text{by CASEAPP}
\end{aligned}$$

(D5): Let $lhs = (\bullet \times Id_D) ; \text{case}$, and $rhs = h_{\cong} ; (Id_{D^n} \times \text{case}) ; \text{case}$, with

$$h_{\cong} = \overline{\lambda x. \langle \pi_1(\pi_1 x), \langle \pi_2(\pi_1 x), \pi_2 x \rangle \rangle}^{(D^n \times D^n) \times D \rightarrow D^n \times (D^n \times D)}.$$

Then (D5) commutes means $lhs = rhs$.

Remember that $\bullet : D^n \times D^n \rightarrow D^n$ is the pairing of all $(Id_{D^n} \times \pi_i^n) ; \text{case}$. Thus

$$\begin{aligned}
\bullet &= \overline{\lambda x. \langle \dots, (\lambda y. t_{\text{case}} (\langle \pi_1 y, \pi_n^i(\pi_2 y) \rangle)) x, \dots \rangle} \\
&= \overline{\lambda x. \langle \dots, t_{\text{case}} (\langle \pi_1 x, \pi_n^i(\pi_2 x) \rangle), \dots \rangle} \\
\bullet \times Id_D &= \overline{\lambda x. \langle \langle \dots, t_{\text{case}} (\langle \pi_1(\pi_1 x), \pi_n^i(\pi_2(\pi_1 x)) \rangle), \dots \rangle, \pi_2 x \rangle} \\
lhs &= \overline{\lambda z. t_{\text{case}} (\langle \dots, t_{\text{case}} (\langle \pi_1(\pi_1 z), \pi_n^i(\pi_2(\pi_1 z)) \rangle), \dots \rangle, \pi_2 z)} \\
&= \overline{\lambda z. \langle \{c_i \mapsto t_{\text{case}} (\langle \pi_1(\pi_1 z), \pi_n^i(\pi_2(\pi_1 z)) \rangle) \}_{i=1}^n \cdot \pi_2 z \rangle} \\
&= \overline{\lambda z. \langle \{c_i \mapsto t_{\text{case}} (\langle \pi_1(\pi_1 z), \pi_n^i(\pi_2(\pi_1 z)) \rangle) \}_{i=1}^n \cdot (\pi_2 z) \rangle} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \langle \{c_j \mapsto \pi_j^n(\pi_1(\pi_1 z)) \}_{j=1}^n \cdot (\pi_n^i(\pi_2(\pi_1 z))) \}_{i=1}^n \cdot (\pi_2 z) \rangle} \\
rhs &= \overline{\lambda z. t_{\text{case}} ((\lambda x. \langle \pi_1 x, t_{\text{case}}(\pi_2 x) \rangle) (\langle \pi_1(\pi_1 z), \langle \pi_2(\pi_1 z), \pi_2 z \rangle \rangle))} \\
&= \overline{\lambda z. t_{\text{case}} (\langle \pi_1(\pi_1 z), t_{\text{case}}(\pi_2(\pi_1 z), \pi_2 z) \rangle)} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z)) \}_{i=1}^n \cdot t_{\text{case}} (\pi_2(\pi_1 z), \pi_2 z) \rangle} \\
&= \overline{\lambda z. \langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z)) \}_{i=1}^n \cdot \langle \{c_j \mapsto \pi_j^n(\pi_2(\pi_1 z)) \}_{j=1}^n \cdot (\pi_2 z) \rangle} \\
&= \overline{\lambda z. \langle \{c_j \mapsto \langle \{c_i \mapsto \pi_i^n(\pi_1(\pi_1 z)) \}_{i=1}^n \cdot \pi_j^n(\pi_2(\pi_1 z)) \}_{j=1}^n \cdot (\pi_2 z) \rangle} \quad (\text{by CASECASE})
\end{aligned}$$

(D6): This diagram commutes if lhs = rhs, with lhs = $\pi_2 ; \downarrow$,

and rhs = $(Id_{D^n} \times \downarrow) ; \mathbf{case}$.

$$\begin{aligned} \text{lhs} &= \overline{\lambda z. (\lambda x. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1) (\pi_2 z)}^{D^n \times \mathbf{1} \rightarrow D} \\ &= \overline{\lambda z. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1}^{D^n \times \mathbf{1} \rightarrow D} \\ \text{rhs} &= \overline{\lambda z. t_{\mathbf{case}} (\pi_1 z, , (\lambda x. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1) (\pi_2 z))}^{D^n \times \mathbf{1} \rightarrow D} \\ &= \overline{\lambda z. t_{\mathbf{case}} (\pi_1 z, , \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1)}^{D^n \times \mathbf{1} \rightarrow D} \\ &= \overline{\lambda z. \{\!\!\{ \mathbf{c}_i \mapsto \pi_i^n(\pi_1 z) \}_{i=1}^n \!\!\} \cdot \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1}^{D^n \times \mathbf{1} \rightarrow D} \\ &= \overline{\lambda z. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1}^{D^n \times \mathbf{1} \rightarrow D} \quad (\text{by CASECASE}) \end{aligned}$$

□

Proposition 3.6. *In the model $\mathcal{M}_{\text{synt}}$, the interpretation of a term t in a context $\Gamma = x_1; \dots; x_k$ is*

$$[t]_\Gamma = \overline{\lambda x. \tilde{t}[x_i := \pi_i^k x]}^{D^k \rightarrow D} \quad (\text{with } x \text{ fresh in } t).$$

Proof: The proof proceeds by structural induction on t . If $t = x_i$ or $t = \mathbf{c}$, we just have to write the definition of $[t]_\Gamma$. If $t = \lambda x_{k+1}. t_0$ or $t = t_1 t_2$, the equation is straightforward from definition of $[t]_\Gamma$ and induction hypothesis. We detail the proof when $t = \{\!\!\{ \theta \} \!\!\} \cdot u$:

$[t]_\Gamma = \langle [\theta]_\Gamma; [u]_\Gamma \rangle; \mathbf{case}$, with $[\theta]_\Gamma = \langle f_1, \dots, f_n \rangle$ where $f_j = [u_j]_\Gamma$ if $\mathbf{c}_j \mapsto u_j \in \theta$, and $f_j = !_{D^k}; \downarrow$ ($= \overline{\lambda x. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1}^{D^k \rightarrow D}$) if $\mathbf{c}_j \notin \text{dom}(\theta)$. So

$$[t]_\Gamma = \overline{\lambda x. t_{\mathbf{case}} (\langle t_\theta x, t_u x \rangle)}^{D^k \rightarrow D}$$

with $\mathbf{case} = \overline{t_{\mathbf{case}}}^{D^n \times D \rightarrow D}$, $[\theta]_\Gamma = \overline{t_\theta}^{D^k \rightarrow D^n}$, and $[u]_\Gamma = \overline{t_u}^{D^k \rightarrow D}$. By induction hypothesis, we can chose $t_u = \lambda x. \tilde{u}[x_i := \pi_i^k x]$, and $t_\theta = \lambda x. (\langle t_1 x, \dots, t_n x \rangle)_n$ with $t_j = \lambda x. \tilde{u}_j[x_i := \pi_i^k x]$ if $\mathbf{c}_j \mapsto u_j \in \theta$, and $t_j = \lambda x. \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1$ if $\mathbf{c}_j \notin \text{dom}(\theta)$.

$$\begin{aligned} \lambda x. t_{\mathbf{case}} , (\langle t_\theta x, t_u x \rangle) &\simeq_{\lambda_{\mathcal{C}}} \lambda x. t_{\mathbf{case}} , (\langle t_1 x, \dots, t_n x \rangle)_n , \tilde{u}[x_i := \pi_i^k x] \\ &\simeq_{\lambda_{\mathcal{C}}} \lambda x. \{\!\!\{ \mathbf{c}_j \mapsto t_j x \}_{j=1}^n \!\!\} \cdot \tilde{u}[x_i := \pi_i^k x] \\ &\simeq_{\lambda_{\mathcal{C}}} \lambda x. \{\!\!\{ \theta \} \!\!\} \cdot u [x_i := \pi_i^k x] \end{aligned}$$

Indeed, $t_j x \simeq_{\lambda_{\mathcal{C}}} \tilde{u}_j[x_i := \pi_i^k x]$ if $\mathbf{c}_j \mapsto u_j \in \theta$, and $t_j \simeq_{\lambda_{\mathcal{C}}} \{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1$ if $\mathbf{c}_j \notin \text{dom}(\theta)$.

Since $D^k \rightarrow D$ is compatible with $\simeq_{\lambda_{\mathcal{C}}}$, $[t]_\Gamma = \overline{\lambda x. \tilde{t}[x_i := \pi_i^k x]}^{D^k \rightarrow D}$. □

B.2 Some rewriting properties

Lemme 3.8.1 ($\lambda_{\mathcal{C}}^-$ reduction on completed terms).

Let t be a defined term. Then, for any term t' ,

$$\tilde{t} \rightarrow_{\lambda_{\mathcal{C}}^-} t' \quad \text{implies} \quad t' = \tilde{t}_0 \quad \text{for some } t_0 \text{ such that } t \rightarrow t_0.$$

Proof: By structural induction on t . First notice that every CASECONS redex present in \tilde{t} corresponds to a CASECONS redex in t , as t is defined. Moreover, $\{\!\!\{ \} \!\!\} \cdot \mathbf{c}_1$ is not reducible so every redex in a sub-term of \tilde{t} corresponds to a redex in a sub-term of t . Also if the reduction $\tilde{t} \rightarrow t'$ is performed in a (strict) sub-term of \tilde{t} , we can immediately conclude with induction hypothesis. So it is sufficient to check the lemma for the five possible reductions in head position $\tilde{t} \rightarrow t'$, which is trivial. □

Lemme 3.8.2 (CASECASE reduction on completed terms).

For any term t, t' ,

$$\tilde{t} \rightarrow_{cc} t' \quad \text{implies} \quad t' \rightarrow_{cc}^* \tilde{t}_0 \quad \text{for some } t_0 \text{ such that } t \rightarrow_{cc} t_0$$

Proof: By structural induction on t . If the CASECASE reduction occurs in a strict sub-term of \tilde{t} then we conclude with induction hypothesis. Otherwise $t = \{\theta\} \cdot \{\phi\} \cdot u$, and $t' = \{\tilde{\theta} \circ \tilde{\phi}\} \cdot \tilde{u}$. Then we take $t_0 = \{\theta \circ \phi\} \cdot u$, since $\tilde{\theta} \circ \tilde{\phi} \rightarrow_{cc}^* \widetilde{\theta \circ \phi}$. Indeed, if $\phi = \{c_i \mapsto u_i / i \in I\}$ then

$$\begin{aligned} \widetilde{\theta \circ \phi} &= \{c_i \mapsto \{\tilde{\theta}\} \cdot \tilde{u}_i / i \in I\} \cup \{c_i \mapsto \{\tilde{\theta}\} \cdot \{\tilde{\phi}\} \cdot c_1 / i \notin I\} \\ \widetilde{\theta \circ \phi} &= \{c_i \mapsto \{\tilde{\theta}\} \cdot \tilde{u}_i / i \in I\} \cup \{c_i \mapsto \{\tilde{\theta}\} \cdot c_1 / i \notin I\} \end{aligned}$$

Also $t' \rightarrow_{cc}^* \tilde{t}_0$. □

Lemma 3.9 (Commutation case-completion/CC-normal form).

For any term t ,

$$\Downarrow(\tilde{t}) = \widetilde{\Downarrow t}.$$

Proof: By induction on the size of the maximal reduction $\tilde{t} \rightarrow_{cc} \Downarrow(\tilde{t})$. If $\tilde{t} = \Downarrow(\tilde{t})$, then \tilde{t} is CASECASE-normal, and so is t (Fact.3.5). Thus $t = \Downarrow t$ and $\tilde{t} = \widetilde{\Downarrow t}$. Otherwise let $\tilde{t} \rightarrow_{cc} t' \rightarrow_{cc}^* \Downarrow(\tilde{t})$. By Lem. 3.8.2, there is a term t_0 such that $t' \rightarrow_{cc}^* \tilde{t}_0$ and $t \rightarrow_{cc} t_0$. Hence $\tilde{t} \rightarrow_{cc}^+ \tilde{t}_0 \rightarrow_{cc}^* \Downarrow(\tilde{t}) = \Downarrow(\tilde{t}_0)$. By induction hypothesis, $\Downarrow(\tilde{t}_0) = \widetilde{\Downarrow t_0}$. Moreover $\Downarrow t_0 = \Downarrow t$, so $(\Downarrow t) = (\Downarrow t_0) = \Downarrow(\tilde{t}_0) = \widetilde{\Downarrow t}$. □

Lemma 3.10. For any terms t, t' , if $t \rightarrow_{\lambda_{\mathcal{C}}} t'$ then there exists a term u such that

$$\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}}^* u \rightarrow_{cc}^* \Downarrow t'.$$

Proof: The proof proceeds by induction on $s(t)$, the structural measure of t defined by

$$\begin{aligned} s(x) &= 1 & s(\lambda x.t) &= s(t) + 1 & s(\{\theta\} \cdot t) &= s(t) \times (s(\theta) + 2) \\ s(c) &= 1 & s(tu) &= s(t) + s(u) & s(\theta) &= \sum_{c \in \text{dom}(\theta)} s(\theta_c) \end{aligned}$$

Notice that this measure decreases with the subterm relation but also with CASECASE reduction ($s(\{\theta\} \cdot \{\phi\} \cdot u) > s(\{\theta \circ \phi\} \cdot u)$ for any θ, ϕ, t). For any term s (or any case-binding θ), s' (*resp.* θ') represents a term (*resp.* a case-binding) such that $s \rightarrow_{\lambda_{\mathcal{C}}} s'$ (*resp.* $\theta_c \rightarrow_{\lambda_{\mathcal{C}}} \theta'_c$ for some $c \in \text{dom}(\theta)$, and $\theta_{c'} = \theta'_{c'}$ for $c' \neq c$)

- If t is an application, either $t = t_1 t_2$ and $t' = t'_1 t_2$ (or $t' = t_1 t'_2$) and we conclude with induction hypotheses, or $t = (\lambda x.t_1)t_2$ and $t' = t_1[x := t_2]$. In that case, $\Downarrow t = (\lambda x. \Downarrow t_1) \Downarrow t_2 \rightarrow_{\lambda_{\mathcal{C}}} (\Downarrow t_1)[x := \Downarrow t_2] \rightarrow_{cc}^* \Downarrow(\Downarrow t_1)[x := \Downarrow t_2]$. Moreover, $\Downarrow(\Downarrow t_1)[x := \Downarrow t_2] = \Downarrow(t_1[x := t_2])$. Thus $\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}} (\Downarrow t_1)[x := \Downarrow t_2] \rightarrow_{cc}^* \Downarrow t'$.
- If t is an abstraction, either $t = \lambda x.t_0$ and $t' = \lambda x.t'_0$ and we conclude with induction hypothesis, or $t = \lambda x.t'x$ with $x \notin \text{fv}(t')$. In that case, $\Downarrow t = \lambda x. \Downarrow t'x \rightarrow_{\lambda_{\mathcal{C}}} \Downarrow t'$.
- If $t = \{\theta\} \cdot x$, then $t' = \{\theta'\} \cdot x$ and we conclude with induction hypothesis.
- If $t = \{\theta\} \cdot c$, then either $t' = \{\theta'\} \cdot c$ and we conclude with induction hypothesis, or $t' = \theta_c$ and $\Downarrow t = \Downarrow \{\theta\} \cdot c \rightarrow_{\lambda_{\mathcal{C}}} \Downarrow \theta_c$.
- If $t = \{\theta\} \cdot t_1 t_2$, then either $t' = \{\theta'\} \cdot t_1 t_2$ and we conclude with induction hypothesis, or $t' = \{\theta\} \cdot t_0$ with $t_1 t_2 \rightarrow_{\lambda_{\mathcal{C}}} t_0$ or $t' = (\{\theta\} \cdot t_1)t_2$.

In the second case, by induction hypothesis there is some u_0 such that $\Downarrow t_1 t_2 \rightarrow_{\lambda_{\mathcal{C}}}^* u_0 \rightarrow_{cc}^* \Downarrow t_0$. Hence

$$\Downarrow t = \Downarrow \{\theta\} \cdot \Downarrow t_1 t_2 \rightarrow_{\lambda_{\mathcal{C}}}^* \Downarrow \{\theta\} \cdot u_0 \rightarrow_{cc}^* \Downarrow \{\theta\} \cdot \Downarrow t_0 \rightarrow_{cc}^* \Downarrow \{\theta\} \cdot \Downarrow t_0.$$

Moreover, every sub-term of $\Downarrow t'$ is in CASECASE normal form, so $\Downarrow t' = \Downarrow \{\!\!\downarrow \Downarrow \theta\!\!\} \cdot \Downarrow t_0$. Thus $\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}}^* \{\!\!\downarrow \Downarrow \theta\!\!\} \cdot u_0 \rightarrow_{cc}^* \Downarrow t'$.

In the last case, $\Downarrow t = \{\!\!\downarrow \Downarrow \theta\!\!\} \cdot (\Downarrow t_1 \Downarrow t_2)$, so

$$\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}}^* (\{\!\!\downarrow \Downarrow \theta\!\!\} \cdot \Downarrow t_1) \Downarrow t_2 \rightarrow_{cc}^* \Downarrow (\{\!\!\downarrow \Downarrow \theta\!\!\} \cdot \Downarrow t_1) \Downarrow t_2 = \Downarrow \{\!\!\downarrow \Downarrow \theta\!\!\} \cdot t_1 \Downarrow t_2.$$

- If $t = \{\!\!\downarrow \theta\!\!\} \cdot \lambda x.t_0$, idem as previous case.
- If $t = \{\!\!\downarrow \theta\!\!\} \cdot \{\!\!\downarrow \phi\!\!\} \cdot t_0$, then either $t' = \{\!\!\downarrow \theta\!\!\} \cdot \{\!\!\downarrow \phi'\!\!\} \cdot t_0$, or $t' = \{\!\!\downarrow \theta\!\!\} \cdot \{\!\!\downarrow \phi\!\!\} \cdot t'_0$, or $t' = \{\!\!\downarrow \theta'\!\!\} \cdot \{\!\!\downarrow \phi\!\!\} \cdot t_0$.
In the first case, write $t_1 = \{\!\!\downarrow \theta \circ \phi\!\!\} \cdot t_0$ and $t'_1 = \{\!\!\downarrow \theta \circ \phi'\!\!\} \cdot t_0$. Remark that $s(t_1) < s(t)$ (since the structural measure decreases by CASECASE-reduction), and that $t_1 \rightarrow_{\lambda_{\mathcal{C}}} t'_1$. By induction hypothesis, there is some u such that $\Downarrow t_1 \rightarrow_{\lambda_{\mathcal{C}}}^* u \rightarrow_{cc}^* \Downarrow t'_1$. Since $\Downarrow t = \Downarrow t_1$ and $\Downarrow t' = \Downarrow t'_1$ we are done.

In the second case, same method but with $t'_1 = \{\!\!\downarrow \theta \circ \phi\!\!\} \cdot t'_0$.

In the last case, write $t = \{\!\!\downarrow \theta\!\!\} \cdot \{\!\!\downarrow \phi_1\!\!\} \cdots \{\!\!\downarrow \phi_k\!\!\} \cdot u_0$, where u_0 is not a case construct (thus $k \geq 1$). Then $\Downarrow t = \{\!\!\downarrow \Downarrow (\theta \circ \psi)\!\!\} \cdot \Downarrow u_0$, with $\psi = \phi_1 \circ (\cdots \circ \phi_k)$, and $\Downarrow t' = \{\!\!\downarrow \Downarrow (\theta' \circ \psi)\!\!\} \cdot \Downarrow u_0$ (since $((\theta \circ \phi_1) \circ \cdots) \circ \phi_k \rightarrow_{cc}^* \theta \circ \psi$).

Let us explicit $\Downarrow t$ and $\Downarrow t'$: $\Downarrow t = \{\!\!\downarrow c \mapsto \Downarrow \{\!\!\downarrow \theta\!\!\} \cdot \psi_c / c \in \text{dom}(\psi)\!\!\} \cdot \Downarrow u_0$
 $\Downarrow t' = \{\!\!\downarrow c \mapsto \Downarrow \{\!\!\downarrow \theta'\!\!\} \cdot \psi_c / c \in \text{dom}(\psi)\!\!\} \cdot \Downarrow u_0$

Remark that $s(\{\!\!\downarrow \theta\!\!\} \cdot \psi_c) \leq s(t)$ (the structural measure decreases by CASECASE-reduction, and preserves the order of sub-term relation), and that $\{\!\!\downarrow \theta\!\!\} \cdot \psi_c \rightarrow_{\lambda_{\mathcal{C}}}^* \{\!\!\downarrow \theta'\!\!\} \cdot \psi_c$. Hence, by induction hypothesis, for each $c \in \text{dom}(\psi)$ there is a term u_c such that $\Downarrow \{\!\!\downarrow \theta\!\!\} \cdot \psi_c \rightarrow_{\lambda_{\mathcal{C}}}^* u_c \rightarrow_{cc}^* \Downarrow \{\!\!\downarrow \theta'\!\!\} \cdot \psi_c$. Thus

$$\Downarrow t \rightarrow_{\lambda_{\mathcal{C}}}^* u \rightarrow_{cc}^* \Downarrow t' \quad \text{for} \quad u = \{\!\!\downarrow c \mapsto u_c / c \in \text{dom}(\psi)\!\!\} \cdot \Downarrow u_0. \quad \square$$